DETAILED ACTION

A. This action is in response to the following communications: Request for Continued Examination filed 01/14/2008.

B. Claims 1-26 remains pending.

Continued Examination Under 37 CFR 1.114

C. A request for continued examination under 37 CFR 1.114, including the fee set forth in 37 CFR 1.17(e), was filed in this application after final rejection. Since this application is eligible for continued examination under 37 CFR 1.114, and the fee set forth in 37 CFR 1.17(e) has been timely paid, the finality of the previous Office action has been withdrawn pursuant to 37 CFR 1.114. Applicant's submission filed on 01/14/2008 has been entered.

Claim Rejections - 35 USC § 102

1. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

2. Claims 1-17, 20-21 and 25-26 are rejected under 35 U.S.C. 102(b) as being anticipated by Macromedia, Extending Dreamweaver,

(http://www.adobe.com/support/dreamweaver/extend.html).

As for independent claim 1, Macromedia teaches a system for controlling user interface features of a web application (pg.7, lines 4-6 of par. 1), the system comprising: a collection of user interface control elements (pg.7, lines 2-4 of par. 1), each control element comprising: a namespace (pg.23, line 3 of par.6 and pg.25, line 1 of par.1); common attributes for defining graphical features of the control element and for associating the control element with the internal state of the core control element (pg.27,first table and pg.208,par.2,line 1); other attributes for defining attributes that affect the intrinsic behavior of the control (pg.216,par.1,line1); and a skin template reference attribute for referencing a skin template (pg.236,par.1); a collection of skin templates comprising extensible markup language based markup (pg.253,last par. and pg.254 first par.) contained as children of a container element (pg.15, first table); and a collection of control element instructions for performing actions associated with the control elements (pg.15,first table; methods), each script associated with a control element (pg.205, lines 1-3 of par.2).

As for dependent claim 2, Macromedia teaches the system as claimed in claim 1, further comprising an initialization function for directing the processing one or more control elements in a document object model, the initialization function comprising

instructions for traversing each node in a document object model (pg.28, par.6) and for searching and calling functions associated with control elements having names following the predetermined naming convention (pg.28, last par.). Namespaces can be emulated to some extent by using a naming convention (page 25, line 1; prefix).

As for dependent claim 3, Macromedia teaches the system as claimed in claim 1, wherein the namespace follows a predetermined naming convention comprises having a constant prefix to the name of the element (page 25, line 1).

As for dependent claim 4, Macromedia teaches the system as claimed in claim 1, wherein the skin template reference attribute comprises a reference to the location of a skin template file (pg.254, par.1).

As for dependent claim 5, Macromedia teaches the system as claimed in claim 1, wherein the control element is associated with an extensible markup language based element (pg.253, last par.).

As for dependent claim 6, Macromedia teaches the system as claimed in claim 5, wherein the control element is a parent of an extensible markup language based element (pg.15, first table).

As for dependent claim 7, Macromedia teaches the system as claimed in claim 5,

wherein the control element is a child of an extensible markup language based element (pg.15, first table).

As for dependent claim 8, Macromedia teaches the system as claimed in claim 2, further comprising: a collection of control attributes for adding to existing regular extensible markup language based elements in a document object model (pg.14, fig.1 and pg.328, par.1), the control attributes following the predetermined naming convention (pg.328; dreamweaver.function ()); and a collection of control attribute instructions for performing actions associated with the collection of control attributes, each instruction associated with a control attribute (pg.329, par.1).

As for dependent claim 9, Macromedia teaches the system as claimed in claim 8, wherein the initialization function contains instructions for traversing each node in the document object model (pg.28, par.6) and for searching and calling functions associated with control elements and control attributes having names following the predetermined naming convention (pg.28, last par.). Namespaces can be emulated to some extent by using a naming convention (page 25, line 1; prefix).

As for dependent claim 10, Macromedia teaches the system as claimed in claim 1, wherein the collection of control elements comprises a markup language (pg.268, par.1).

As for dependent claim 11, Macromedia teaches the system as claimed in claim 1, wherein the common attributes comprise state attributes for specifying the identification of a <state> child element of the control element (pg.15, table; nodes and pg.26 and 27,tables; states).

As for dependent claim 12, Macromedia teaches the system as claimed in claim 1, wherein the common attributes comprise one or more of: an identification attribute for referencing the control element (pg.329, par.1); a label attribute for associating text control (pg.25, table; name); a height attribute for specifies the height of the control element (pg.25, table; size); a disabled attribute for specifying whether the control element is disabled and cannot be used (pg.26, table; state); a state hover attribute for specifying the identification of a <state> child element of the control element, the state hover attribute used to override the appearance of a hover state as defined in a skin of the control element (pg.16, table; image and pg.236, par.1); a state focus attribute for specifying the identification of a<state> child element of the control element, the state focus attribute used to override the appearance of a focus state as defined in a skin of the control element (pg.17,table;select and pg.236, par.1); a state up attribute for specifying the identification of a <state> child element of the control element, the state up attribute used to override the appearance of an up state as defined in a skin of the control element (pg.16, table; image and pg.236, par.1); a state down attribute for specifying the identification of a <state> child element of the control element, the state down attribute used to override the appearance of a down state as defined in a skin of

the control element (pg.16, table; image and pg.236, par.1); a state hit attribute for specifying the identification of a <state> child element of the control element, the state hit attribute used to override the appearance of a hit state as defined in a skin of the control element (pg.16, table; checkbox and pg.236, par.1); a state disabled up attribute for specifying the identification of a <state> child element of the control element, the state disabled up attribute used to override the appearance of a disabled up state as defined in the skin of the control element (pg. 26,table; state and pg.236, par.1); and a state disabled down attribute for specifying the identification of a <state> child element of the control element, the state disabled down attribute used to override the appearance of a disabled down state as defined in a skin of the control element (pg. 26,table; state and pg.236, par.1).

As for dependent claim 13, Macromedia teaches the system as claimed in claim 12, wherein the set of control elements comprises one or more of: It is evident that the following: button, combo box, list box, list view, context menu, item, text box, slider, scrollbar and spin dial, are taught by Macromedia and thus only one will be analyzed in detail.

a dsvg:button control element for defining a control that is clicked to trigger an action (pg.16, object-button, event-onClick), the dsvg:button control element comprising: a namespace following the predetermined naming convention (pg.14, line1 of last par.);

the common attributes (pg.16,table 1-button); other attributes comprising: a toggle attribute for specifying whether the button is a toggle or a sticky button (pg.16,table 1, button –event – onClick); a group attribute for specifying the name of a group to which the button control element belongs (pg.15, table 1, alltags/elements); and a checked attribute for specifying whether the button control element is down/checked or up/unchecked (pg.16, table 1, button); a skin template reference attribute for specifying the location of a control element skin template (pg.254, par.1), the skin template reference settable to a uniform resource index (pg.254, par.1); and a customizable skin template comprising scalable vector graphics markup contained as children of a container element (pg.36, par.3 and (pg.15, first table);

As for dependent claim 14, Macromedia teaches a system for controlling user interface features of a web application (pg.7, lines 4-6 of par. 1), the system comprising: a collection of control element instructions for performing actions associated with the control elements, each instruction associated with a control element (pg.329, par.1); and an initialization function for directing the processing of one or more control elements in a document object model (pg.28, par.6 and last par.).

As for dependent claim 15, Macromedia teaches the system as claimed in claim 14, further comprising a collection of skin templates comprising extensible markup language based markup (pg.253, last par. and pg.254 first par.) contained as children of a container element (pg.15, first table).

As for dependent claim 16, Macromedia teaches a method of controlling user interface features of a web application, the method comprising the steps of: searching for a designated user interface control element in a document object model; and calling a script associated with the designated control element (pg.28, par. "traversing nodes" and "getting node data").

As for dependent claim 17, Macromedia teaches the method as claimed in claim 16, wherein the step of searching includes the steps of: traversing each node in the document object model (pg.28, par.8); and determining whether an element has a name which follows a designated naming convention (pg.28, par.8; node.name).

As for dependent claim 20, Macromedia teaches the method as claimed in claim 16, further comprising the steps of: searching for a designated attribute in an element in a document object model; and calling a script associated with the designated attribute (pg.28, last par.).

As for dependent claim 21, Macromedia teaches the method as claimed in claim 20, wherein the step of searching for a designated attribute comprises the steps of: searching attributes of an element in a document object model; determining whether an element attribute has a name which follows a designated naming convention (pg.19; "getElementsByTagName (tagName)"). Obtaining attributes from tags, the tags will also

act as a prefix for elements, which belong to a certain attribute(s).

As for dependent claim 25, Macromedia teaches a method of controlling user interface features of a web application (pg.9, lines 1-2 of par.2), the method comprising the steps of: adding a behavior element as a child of a user interface control element (pg.216, "dom.addBehavior ()"); receiving an event which is equal to an event attribute setting in the behavior element (pg.217, "dom.getBehavior()", where onClick attribute is equal an event to opening a script event this event is received); and calling a script associated with the behavior element (pg.223,fig.1).

As for independent claim 26, Macromedia teaches a method of creating a customizable user interface control element having expected behaviors (pg.10, par.1 and 5), the method comprising the steps of: categorizing user interface controls into fundamental core controls (pg.31, line 2-3 of par.1); determining variations of a core controls (pg.16, table 1); determining common attributes of the core control (pg.16, table 1); determining fundamental states for the core control (pg.16, table 1); determining how to allow for absolute positioning of objects the core control (pg.35, par.1); determining how to allow for absolute customization of appearance of the core control; assigning a reference link to the core control (pg.36, par.3-4); determining templates for skins to allow for the absolute customization of appearance of the core control (pg.254, par.1); determining how to associate behaviors to the core control (pg.36, par.4); and creating a core control element (pg.36, par.1).

Claim Rejections - 35 USC § 103

3. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

4. This application currently names joint inventors. In considering patentability of the claims under 35 U.S.C. 103(a), the examiner presumes that the subject matter of the various claims was commonly owned at the time any inventions covered therein were made absent any evidence to the contrary. Applicant is advised of the obligation under 37 CFR 1.56 to point out the inventor and invention dates of each claim that was not commonly owned at the time a later invention was made in order for the examiner to consider the applicability of 35 U.S.C. 103(c) and potential 35 U.S.C. 102(e), (f) or (g) prior art under 35 U.S.C. 103(a).

5. Claims 18-19 and 22-24 are rejected under 35 U.S.C. 103(a) as being unpatentable over Macromedia in view of Cain (US 6,014,138).

As for dependent claim 18, Macromedia teaches the method as claimed in claim 16, wherein the step of calling a script includes the steps of (note claim 16 above). Macromedia does not specifically mention dynamically generating a function name,

passing an object, retrieving the attributes or performing a function having a generated name. However in the same field of endeavor Cain teaches dynamically generating a function name associated with the designated element (col. 12, lines 12-13); passing an object associated with the designated element as a parameter of the generated function (fig. 4H); retrieving the attributes of the object (col.12, lines 29-31); and performing a function stored in memory having the generated function name (fig.4H-I). It would have been obvious to one of ordinary skill in the art at the time of the invention to combine the method of Cain into the method of Macromedia; this is true because Cain's methods solve the same problem of Macromedia's of custom building a graphical operator interface (col.1, lines 23-25).

As for dependent claim 19, Macromedia teaches the method as claimed in claim 18, wherein the step of dynamically generating includes the steps of (note claims 16 and 18 above). Macromedia teaches determining if the name of the designated element contains a designated prefix (pg.25, par.1). Macromedia does not specifically mention generating a function name; assigning an object or assigning predetermined instructions. However in the same field of endeavor Cain teaches generating a function name comprising of the name of the designated element (col.12, lines 12-13); assigning an object associated with the designated element as the parameter of the function (fig.4H); and assigning predetermined instructions of the designated element as steps for the function to perform (fig.4H-I). It would have been obvious to one of ordinary skill in the art at the time of the invention to combine the method of Cain into the method of

Macromedia; this is true because Cain's methods solve the same problem of Macromedia's of custom building a graphical operator interface (col.1, lines 23-25).

As for dependent claim 22, Macromedia teaches the method as claimed in claim 21, wherein the step of calling a script includes the steps of: determining if the name of the designated attribute contains a designated prefix (pg.19, note claim 21 above); Macromedia does not specifically mention generating a function name, assigning an attribute ,object or predetermined instructions. However in the same field of endeavor Cain teaches generating a function name comprising of the name of the designated attribute (col. 12, lines 12-13, i.e. "click"); assigning an object associated with the designated attribute as the parameter of the function name (fig.4H); and assigning predetermined instructions of the designated attribute as steps for a function having the function name to perform (fig.4H-I). It would have been obvious to one of ordinary skill in the art at the time of the invention to combine the method of Cain into the method of Macromedia; this is true because Cain's methods solve the same problem of Macromedia's of custom building a graphical operator interface (col.1, lines 23-25).

As for dependent claim 23, Macromedia teaches the method as claimed in claim 20 (note the discussion of claim 20 above). Macromedia does not specifically mention in the steps of calling a script. However in the same field of endeavor Cain teaches the steps of calling a script comprising: dynamically generating a function name associated with the designated attribute (col. 12, lines 12-13); passing an object associated with the

designated attribute as a parameter of the generated function name (fig.4H); receiving the attributes of the object (col.12, lines 29-31); and performing a function stored in memory having the generated function name (fig.4H-I). It would have been obvious to one of ordinary skill in the art at the time of the invention to combine the method of Cain into the method of Macromedia; this is true because Cain's methods solve the same problem of Macromedia's of custom building a graphical operator interface (col.1, lines 23-25).

As for dependent claim 24, Macromedia teaches the method as claimed in claim 18, wherein the step of dynamically generating includes the steps of (note claims 16 and 18 above). Macromedia teaches determining if the name of the designated attribute contains a designated prefix (pg.25, par.1). Macromedia does not specifically mention generating a function name; assigning an object or assigning predetermined instructions; However in the same field of endeavor Cain teaches generating a function name comprising of the name of the designated attribute (col.12, lines 12-13); assigning an object associated with the designated attribute as the parameter of the function (fig.4H); and assigning predetermined instructions of the designated attribute as steps for the function to perform (fig.4H-I). It would have been obvious to one of ordinary skill in the art at the time of the invention to combine the method of Cain into the method of Macromedia; this is true because Cain's methods solve the same problem of Macromedia's of custom building a graphical operator interface (col.1, lines 23-25).

(Note:) **It is noted that any citation to specific, pages, columns, lines, or figures in the prior art references and any interpretation of the references should not be considered to be limiting in any way. A reference is relevant for all it contains and may be relied upon for all that it would have reasonably suggested to one having ordinary skill in the art. In re Heck, 699 F.2d 1331, 1332-33, 216 USPQ 1038, 1039 (Fed. Cir. 1983) (quoting In re Lemelson, 397 F.2d 1006,1009, 158 USPQ 275, 277 (CCPA 1968)).**

Response to Arguments

Applicant's arguments filed 01/14/2008 have been fully considered but they are not persuasive.

After careful review of the amended claims (given the broadest interpretation) and the remarks provided by the Applicant along with the cited reference(s) the Examiner does not agree with the Applicant for at least the reasons provided below:

A1. Applicant argues that Macromedia does not allow a control element to be specified in an extensible mark-up language through associating the required functionality with the element using an attribute of the control element.

R1. Examiner does not agree, Macromedia points out on page 25 the functionality of the dreamweaver prefix MM: which acts as a namespace to allow already created functions and functions which are not created yet by the user to distinguish among html code thus the MM: prefix is a namespace to a plurality of function controls to be used at the users discretion.

A2. *Applicant argues that* Dreamweaver *does not control user interface features of a web application (page 21).*

R2. Examiner does not agree. Dreamweaver alone is a web application for building web applications, as known in the art http://en.wikipedia.org/wiki/Dreamweaver. The reference cited which only depicts the customizability of a web applications (Dreamweaver) wherein stated on page 7 being able to customize the API of Dreamweaver.

A3. *Applicant argues that the extensions listed by Dreamweaver are only for the use of making HTML.*

R3. Examiner believes the applicant is wrong, in chapter 1 it is explained that the user is modifying the Dreamweaver interface (panels, menus, dialog boxes, html formats), which of course is done by modifying a Document Object Model, which is writing in a scripting language "JavaScript". Of course those skilled in the art that any scripting language would have been an obvious variant.

A4. *Applicant argues that Dreamweaver cannot change the graphical appearance of an extension (page 21).*

R4. Examiner disagrees and points attention to page 10, which describes the user changing the API graphically in Dreamweaver.

A5. *Applicant argues that CSS styles cannot define visual appearance of control elements because they are not defined in HTML (page 22).*

R5. Examiner disagrees, on page 13 Dreamweaver begins to explain how controls are writing in JavaScript, which are in HTML and the like.

A6. *Applicant argues that Dreamweaver does not disclose manipulating a tree control (page 22).*

R6. Examiner disagrees as Dreamweaver clearly points out manipulation of a DOM tree in chapter 2 and 20 among others.

A7. *Applicant argues that Dreamweaver fails to associate instructions with control elements where the instructions are for performing actions associated with the control elements (page 23).*

R7. Examiner does not agree, if the user creates a control for performing a function, therefore associated instructions are contained for the control (see page 24, and chapter 20).

A8. *Applicant argues that Dreamweaver does not disclose a collection of skin templates (page 24).*

R8. Examiner does not agree Dreamweaver makes use of skin templates as depicted on pages 16, 236, 253, 254 and chapters 2, 12, 15, 17 and 20.

A9. *Applicant argues Dreamweaver fails to call a script associated with a control element (page 24).*

R9. Examiner does not agree, Dreamweaver makes use of JavaScript for handing the controls of the API (chapters 2 and 17).

A10. *Applicant argues Dreamweaver does not teach behaviors in such that calling an associated script (page 26).*

R10. Examiner does not agree note page 10 of Dreamweaver where calling a script by action of a function is executed, is described. Note chapter 17.

A11. *Applicant argues Cain does not teach dynamically generating a function name, that Cain teaches automatically assigning a generic function name (page 27-29).*

R11. Examiner believes the applicants contradict themselves. Dynamically and automatically is in the same meaning. Cain teaches creating a function name automatically, the same as dynamically. Also the name assigned to the function relates to the function ("button03") for a function of a button on a graphical user interface.

Conclusion

The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

Inquires

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Nicholas Augustine whose telephone number is 571-270-1056. The examiner can normally be reached on Monday - Friday: 7:30- 5:00.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Weilun Lo can be reached on 571-272-4847. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/Nicholas Augustine/
Examiner
April 10, 2008

/Ba Huynh/
Primary Examiner, Art Unit 2179